

Abstract

The paper addresses the k -tangle enumeration problem. We introduce a notion of cascade diagram for k -tangle projections. An effective enumeration algorithm for projections is proposed based on cascade representation. Tangles projections with up to 12 crossings are tabulated. We provide also pictures of alternating k -tangles with 5 crossing or less.

Classification of k -tangle projections using cascade representation

A. Bogdanov, V. Meshkov, A. Omelchenko, M. Petrov
St Petersburg Polytechnic University, St Petersburg, Russia

July 20, 2010

Introduction

Tangles (see Figure 1) were introduced by John Conway [1] as an instrument for the description and classification of knots and links. According to up-to-date definitions [2], Conway's tangles are 2-tangles. A natural generalization of 2-tangles are k -tangles or k -string tangles (Figure 1b), where k is one-half of the number of “legs”.

As in the case of knots and links we can set the classification problem for k -tangles. This problem is only solved for some specific subclasses of 2-tangles [3, 4] (e.g. rational tangles) which are most-used for studying knots. The general case of k -tangles is interesting in itself. Furthermore it is possible to use k -tangles for representing knots and links in a “thickened” closed oriented surface of arbitrary genus (virtual links [8, 9]). So k -tangle classification can be a basis for classifying virtual links.

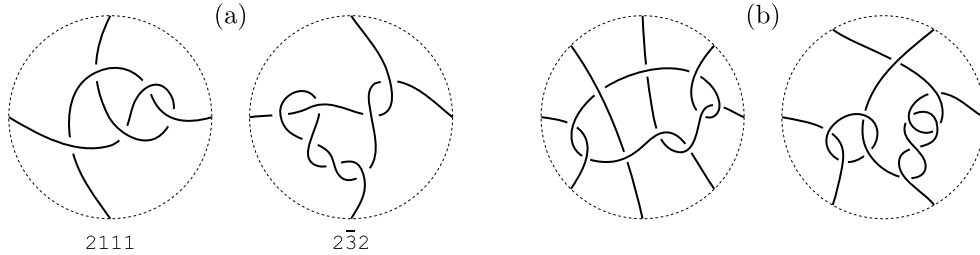


Figure 1 2-tangles (Conway's notation) (a), k -tangles in general case (b)

In this paper we introduce a notion of cascade diagram and propose a classification algorithm for k -tangles based on the cascade representation. We represent cascade diagrams by a code that is almost as compact as DT-code (Dowker–Thistlethwaite) for knots [12]. This cascade code actually is a recipe how to draw the projection so there is no problem with verification of realizability of a given code.

The first step in most of knots enumeration algorithms is enumeration of projections. Having the full set of projections with n crossings, it is easy to generate all the knots (links, tangles) by arranging under- and over-crossing in each projection by all 2^n possible ways [11].

Projections enumeration is related directly to enumeration of alternating knots (links, tangles). For any projection it is possible to arrange crossings in such a way that “under” and “over” will alternate when traveling along each component of the link (tangle). It is known that a (reduced) alternating diagram of a given link has minimal number of crossings (Tait conjecture). Any projection determines uniquely (up to mirror symmetry) an alternating knot (link, tangle). However, the set of projections with n crossings is wider than the set of alternating knots (links, tangles): all alternating diagrams obtained from a given one by a series of flips (Figure 2) correspond to the same knot (link, tangle).

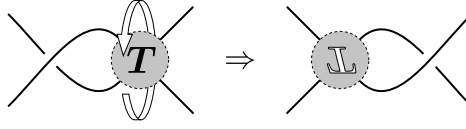


Figure 2 A flip

In the paper we will concentrate first of all on classification of tangle projections.

1 Basic definitions

Definition 1. A proper embedding of a disjoint union of k arcs and (possibly) some number of circles into the standard 3-ball B^3 is called a k -tangle. The $2k$ endpoints are fixed at the points $(\cos \pi i/k, \sin \pi i/k, 0)$, $i \in \{0, 1, \dots, 2k-1\}$ on the boundary sphere (Figure 3). k -tangle is considered up to the dihedral symmetry group D_{2k} .

Definition 2. Tangles T_1 and T_2 are *equivalent* if there is an isotopy i.e. a smooth family $f_t: B^3 \rightarrow B^3$, $t \in [0, 1]$ of smooth maps that keep the boundary sphere fixed such that $f_0 = \text{id}$ and tangles T_2 and $\tilde{T}_1 = f_1(T_1)$ are congruent up to dihedral group transformations.

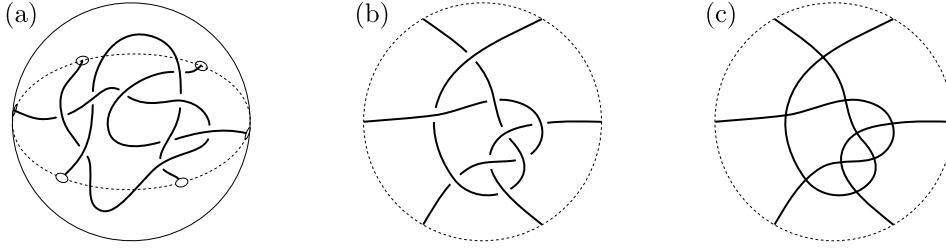


Figure 3 A 3-tangle, its diagram and projection

Along with the equivalence defined above a weaker equivalence relation is also considered that allows the legs of the tangle to be moving on the boundary sphere [5].

Definition 3. Tangles T_1 and T_2 are *weakly equivalent* if there is a smooth family $f_t: B^3 \rightarrow B^3$, $t \in [0, 1]$ of smooth homeomorphisms on the ball B^3 such that $f_0 = \text{id}$ and $f_1(T_1) = T_2$.

In [3] 2-tangles with 7 crossings or less are classified up to weak equivalence.

For graphical representation of tangles conventional *diagrams* are used which are non-singular projections onto the equatorial disk $x^2 + y^2 \leq 1$, $z = 0$ with additional information at each double point about which “thread” is above the other one (Figure 3b). We are more interested here in projections in themselves (Figure 3c).

Definition 4. A tangle projection (diagram) is called *composite*, if there is a smooth closed curve inside the boundary circle that intersects the projection exactly twice and encloses at least one crossing (Figure 4a,b). Otherwise the projection is called *prime*.

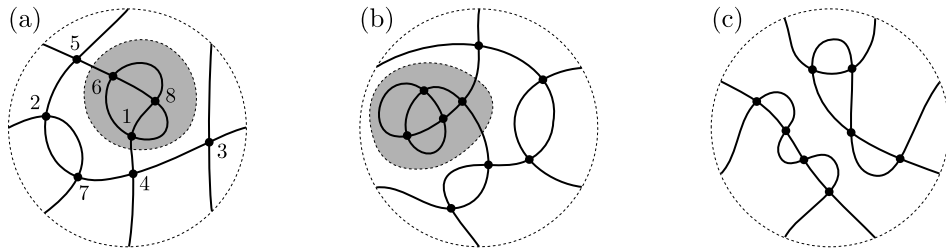


Figure 4 Composite (a,b) and non-connected (c) projections

As in the case of knots and links we are interested in classification of prime projections only. Moreover, it is natural to consider only connected projections, i.e. such that it is possible to get from any crossing to any other moving along the arcs of the projection. An example of non-connected projection is shown in Figure 4c.

We will use the terminology of graph theory for description of projections. In particular, the crossings of projection will be called *vertices*, and arcs will be called *edges*. The set of vertices will be denoted by V , and the set of edges will be denoted by E .

Finally, for the following we need two more definitions.

Definition 5. A crossing (vertex) is called *boundary crossing* (respectively, *vertex*) if some of the incident edges are connected with the boundary circle.

Definition 6. A *cut-crossing* (vertex) is a crossing (vertex) that if removed (together with the incident edges) produces a non-connected graph.

For example, the projection Figure 4a contains five boundary crossings 2, 3, 4, 5, 7. The crossing number 4 is a cut-crossing.

2 Cascade diagram of projection

Any prime connected projection can be represented as a *cascade diagram* which is defined in the following way. Let us construct a system of nested areas in the disk so that there is exactly one vertex in each annular layer (Figure 5), and a connected sub-projection is contained inside each area. Of course, it always can be done in many ways.

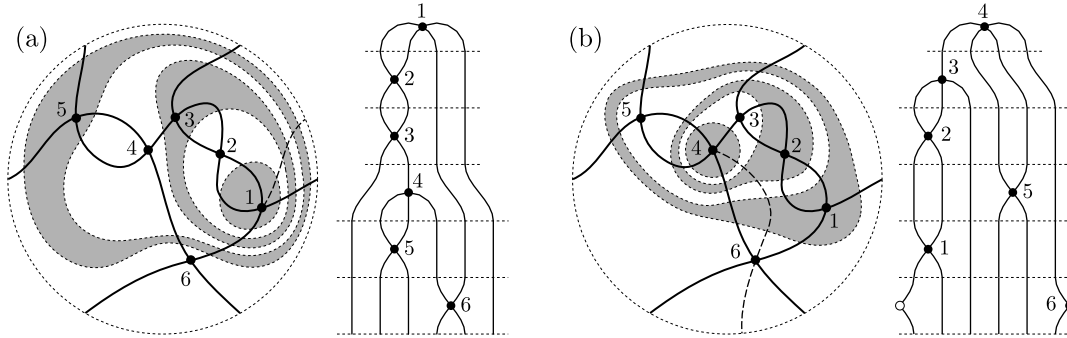


Figure 5 Cascade diagram construction

Let us cut the disk from the central vertex (vertex 1 in Figure 5) to the boundary circle so that the cutting line intersects the boundary of each area exactly once (wide-dashed lines in Figure 5). It is allowed the cut to dissect the projection at vertices only (see Figure 5b). Opening the cut, we obtain a representation of the projection as a falling cascade (Figure 5). We will call this representation *cascade diagram* of the projection.

There is only one vertex at each level of the cascade diagram. Edges incident to this vertex intersect upper and/or lower boundaries of the level. Only five configurations are possible, that correspond to the patterns

$$\overline{\cap}, \emptyset, \overline{\cup}, \Psi, \Sigma.$$

The $\overline{\cap}$ pattern, which is always the starting pattern, cannot be located at lower levels because of connectivity of the projection and by construction. Moreover, let us prove the following statement:

Theorem 1. Any prime projection has a cascade diagram free of the \emptyset pattern; any vertex can be chosen as the starting vertex of such cascade diagram.

We shall prove the theorem by constructing an explicit recurrent algorithm that allows us to construct a cascade diagram required.

Lemma. *Any prime projection has at least two non-cut boundary crossings.*

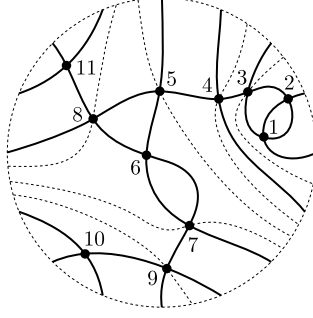


Figure 6 On the proof of the Lemma

Proof. Through each cut-crossing draw a chord (a simple curve with end points on the boundary circle) that nowhere else intersects the projection (Figure 6). It is always possible by Def. 6, and can be done such a way that the chords do not cross each other. Obviously, there are at least two “outermost” chords (e. g. the chords passing through the crossings 3, 8, 9 in Figure 6). Any of these chords cuts a “lune” (a part of the disk not containing cut-crossings). But by the cut-crossing definition any lune should contain at least one crossing; moreover, there is at least one boundary crossing in each lune (contrary is possible only for composite projections of type Figure 4b). So, for each lune there is at least one non-cut boundary crossing. This completes the proof. \square

Proof of the Theorem 1. Set an arbitrary crossing to be the starting crossing of the cascade. We will build the cascade diagram in bottom-up direction. Choose one of non-cut boundary crossings that is not the starting one (it is always possible by the Lemma). Put this crossing on the lower n -th level (see Figure 5). The residual part of the projection is again a connected prime projection; it has $n-1$ crossings. By the Lemma this projection has at least two non-cut boundary crossings. So again we can choose a crossing different from the starting one and put it on the $(n-1)$ -th level. Repeat this procedure until all the crossings (except the starting one) will arrange; so we construct a cascade diagram required. The theorem is proved. \square

Thus, any projection can be represented by a cascade diagram containing (besides the starting pattern $\overline{\text{A}}$) the patterns $\overline{\text{B}}$, $\overline{\text{C}}$, $\overline{\text{D}}$ only; it is easily shown that each of these patterns is necessary.

3 Cascade code

Let us code levels of cascade diagram starting from the top. The pattern $\overline{\text{A}}$ is always on the first level so we will not include it in the code. For other levels we will use the following coding scheme. Associate a pair (α_i, m_i) with i -th level of the cascade diagram. Here α_i denotes one of the patterns $\overline{\text{B}}$, $\overline{\text{C}}$, $\overline{\text{D}}$, and an integer m_i determines the shift of the pattern α_i with respect to the pattern α_{i-1} on the previous level. We should choose a reference point for each pattern to determine the shift m_i : let settle the origin on the left leg for the $\overline{\text{D}}$ pattern, and on the central leg for the patterns $\overline{\text{B}}$ and $\overline{\text{C}}$. The shift $m_i \in 0, 1, \dots, k_{i-1}-1$ counts in counterclockwise direction (in left-right direction on cascade diagram).

So a diagram with n crossings is coded by an ordered set of $n-1$ pairs:

$$C = \{(\alpha_2, m_2), \dots, (\alpha_n, m_n)\}$$

(recall that the code does not include the starting pattern $\overline{\text{A}}$). For example, the code

$$C = \{(1, 0), (0, 5), (0, 3), (0, 3), (0, 5)\}$$

corresponds to the cascade diagram Figure 5b.

Cascade code for k -tangles with n crossings is almost as compact as the DT-code for knots with n crossings [12, 13]: we need only 2 bits per crossing additionally for storing α_i . It is properly that the length of cascade code only depends on the crossing number; it does not depend on the number of tangle components (and therefore on the number of legs).

Cascade code actually is a recipe how to draw the projection, so we have no problems with checking of the code drawability (it is well known that it is not an easy task for DT-code). The last property allows us to use the cascade code effectively for generation and enumeration of tangle projections with given number of crossings.

Each projection has a lot of cascade representations so the equivalence problem for cascade codes occurs. In order to resolve this problem it is necessary to define a *canonical cascade code* which should correspond uniquely to a given projection. Such code can be constructed in different ways. We will define it in such a way that the important *nesting property* will be satisfied.

If $C_n = \{(\alpha_2, m_2), \dots, (\alpha_n, m_n)\}$ is a canonical cascade code, then any initial segment $C_i = \{(\alpha_2, m_2), \dots, (\alpha_i, m_i)\}$, $2 \leq i \leq n-1$, of the code is again the canonical cascade code of the corresponding sub-tangle with i crossings. (*)

In order to do define canonical cascade code, we will at first define an auxiliary invariant code of projection.

4 Invariant root-code of projection

Definition 7. A *root* of a projection is a triple $r = (v, e, f)$ where $v \in V$ is a vertex of the projection, $e \in E$ is one of four edges incident to v , f is one of two faces incident to e (see examples in Figure 7). v , e , and f is called *root-vertex*, *root-edge*, and *root-face* respectively.

The location of the root-face relative to the root-edge and the root-vertex determines naturally a rotation direction (clockwise or counter-clockwise) about the root-vertex which we will call the *labeling direction*.

Vertices of the projection can be uniquely labeled as a root fixed. As an example the following algorithm may be used:

0. set the number of the root-vertex as 1;
 number vertices adjacent to the root-vertex
 in the labeling direction starting from the root-edge;
1. among the labeled vertices find the vertex q
 with the smallest number that has unlabeled neighbors;
2. assign the numbers to unlabeled vertices around the vertex q
 in order determined by the labeling direction,
 starting from the first unlabeled edge;
3. if there are still unlabeled vertices return to the step 1.

The vertices of the projection in Figure 7a,b are labeled using the algorithm described above.

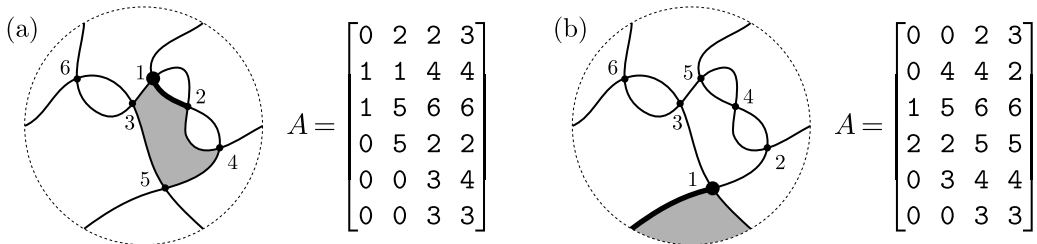


Figure 7 Two vertex labelings induced by different roots, the adjacency lists; the canonical labeling (b)

During the labeling algorithm execution an *adjacency list* A_r is constructed. The i -th line of the adjacency list contains labels of the vertices adjacent to the i -th vertex listed in the labeling direction. For the boundary vertices we substitute 0 at the positions corresponded to the boundary edges (see Figure 7).

The order of numbers in each line of the list is determined up to a cyclic shift. Let us choose the lexicographically minimal sequence from the four possible variants. (The line $a = [a_1, \dots, a_n]$ is *lexicographically less* than line $b = [b_1, \dots, b_n]$ if there is j such that $a_j < b_j$ and $a_i = b_i$ for any $i < j$.)

Let us agree to write the adjacency list A_r in one line. For example, the root in Figure 7a determines the adjacency list

$$A_r = [0 \ 2 \ 2 \ 3 \ 1 \ 1 \ 4 \ 4 \ 1 \ 5 \ 6 \ 6 \ 0 \ 5 \ 2 \ 2 \ 0 \ 0 \ 3 \ 4 \ 0 \ 0 \ 3 \ 3].$$

Definition 8. A single-line adjacency list A_r constructed as described above is called the *root-code* corresponding to a given root r .

Thus as a root fixed the corresponding root-code is defined uniquely; in turn a root-code determines uniquely the corresponding projection (up to rotations and reflections).

We can define a root of a given projection in $8n$ ways. Therefore, there are $8n$ generally different root-codes of the projection; so in order to define an invariant code we need a rule that allows to choose a unique root-code from this set. The root-code constructed by that rule we will call canonical and will denote as $\text{rcode}(\cdot)$.

Definition 9. Root-code $\text{rcode}(\cdot)$ is called an *invariant root-code* if it is determined uniquely by a given projection:

$$\text{rcode}(P_1) = \text{rcode}(P_2) \iff P_1 \cong P_2.$$

For example one of the simplest rules (but in the same time one of the most inefficient) is the following one: define the invariant root-code as the lexicographic minimum among all $8n$ codes.

In order to construct a canonical cascade code satisfying the nesting property (*) we need the root-vertex to be *boundary* and *non-cut*. It is obvious that a root-code will be lexicographically minimal only if the root-vertex has *maximum number of legs*: in this case there will be maximum number of starting zeroes in the root-code. For the projection in Figure 7a even these two requirements reduce the search set to two root-vertices 2 and 6.

We can reduce the number of compared codes some more, for example using a face-code notion. Let us associate an integer with each boundary face — the number of projection edges adjacent to the face. Then we can write these numbers for all boundary faces in order defined by the labeling direction as a string of length $2k$ (number of legs). The *face-code* corresponding to a given (boundary) root is such a sequence that the starting number corresponds to the root-face. For example the face-code for the root in Figure 7b is $[2 \ 3 \ 4 \ 4 \ 2 \ 4]$.

Thus, we select an invariant subset in the set of roots

$$R = \left\{ r \left| \begin{array}{l} \text{root-vertex is a boundary vertex,} \\ \text{root-vertex is a non-cutting vertex,} \\ \text{face-code is lexicographically minimal} \end{array} \right. \right\}. \quad (1)$$

Now let us define *invariant root-code* $\text{rcode}(\cdot)$ as the lexicographically minimal root-code on the subset R :

$$\text{rcode}(T) = \text{lexmin}_{r \in R} \{A_r(T)\}.$$

Definition 10. The root defining the invariant root-code we will call *canonical root*.

The subset R for the projection in Figure 7 contains only one root, namely, the root depicted in Figure 7b; the invariant root-code is the following:

$$\text{rcode}(P) = [0 \ 0 \ 2 \ 3 \ 0 \ 4 \ 4 \ 2 \ 1 \ 5 \ 6 \ 6 \ 2 \ 2 \ 5 \ 5 \ 0 \ 3 \ 4 \ 4 \ 0 \ 0 \ 3 \ 3].$$

It is important to notice that the average number of elements in the subset R (among all the projections with given number of crossings) tends to 1 as crossing number n increases. This is because the fraction of symmetrical face-codes tends to 0. Consequently, in case of large n with probability close to 1 the canonical root can be found without calculation of the canonical root-code.

5 Canonical cascade code

Now, using invariant root-code described above, we can define a canonical cascade code satisfying the nesting property (*).

Consider a tangle projection T_n with n vertices. We will construct a canonical cascade diagram (and canonical cascade code) from bottom to top using a procedure similar to that used in the proof of the Theorem 1; let us describe it omit some details.

Find the canonical root-vertex of the projection P_n (construct the canonical root-code if it is necessary); put this vertex on the n -th level of the cascade being constructed. In doing so, we obtain a tangle projection P_{n-1} (with $n-1$ crossings) corresponds to the rest part of the projection. For the projection P_{n-1} we can again find the canonical root-vertex; and so the $(n-1)$ -th level of the cascade diagram will defined. Continuing, we construct a uniquely determined cascade diagram and, in the same time, a unique sequence of nesting tangle projections:

$$\overbrace{\text{---}}^{\text{---}} = P_1 \leftarrow P_2 \leftarrow \cdots \leftarrow P_{n-1} \leftarrow P_n. \quad (2)$$

The cascade code built using the described algorithm we will call *canonical cascade code* of projection.

The canonical cascade for the projection Figure 5 is shown in Figure 5b; the corresponding canonical cascade code is the following:

$$\begin{array}{c} \{(0, 0), (0, 0), (1, 1), (0, 5), (0, 2)\} \\ \underbrace{\hspace{1.5cm}}_{P_2} \\ \underbrace{\hspace{1.5cm}}_{P_3} \\ \underbrace{\hspace{1.5cm}}_{P_4} \\ \underbrace{\hspace{1.5cm}}_{P_5} \\ \underbrace{\hspace{1.5cm}}_{P_6} \end{array}$$

The sequence (2) determines a “genealogy” of the projection. For the projection Figure 5 we obtain the series depicted in Figure 8.

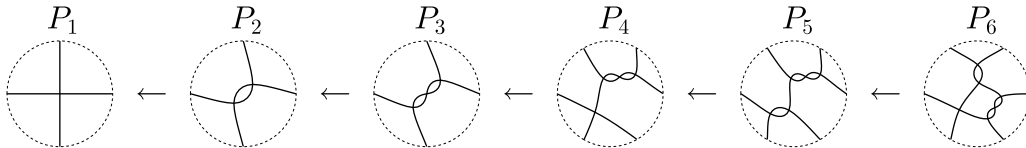


Figure 8 The genealogy of the projection Figure 5

Thus, for a given projection the parent-projection is uniquely defined by the canonical cascade code. We will use this fact in the next section to construct an enumeration algorithm for k -tangle projections.

6 Enumeration algorithm

Algorithms of knots and links classification usually include two stages: 1) generation, and 2) sifting. The main objective of the generation stage is to obtain (using some representation of knot projections, e.g. DT-code) a set that includes all the projections with given number of crossings. As a rule, the obtained set is redundant, because there are lots of different representations of the same projection. These duplicates must be found using invariants and eliminated in the second stage of the algorithm, in the sifting stage. The effectiveness of the classification algorithm depends on the amount of redundancy and on complexity of the equivalence checking procedure.

Recent classification algorithms [11, 13] realizing the generation stage based on the succession principle: knots with $n+1$ crossings are generating from the (sifted) set of knots with n crossings. For example, in the algorithm [13, 14, 15] each knot with n crossings generates a set of child knots with $n+1$ crossings using local operations like shown in Figure 9.



Figure 9 Local operations increasing the crossing number by 1 [13]

Such approach keeps the amount of duplicates relatively small and prohibits generation of non-drawable representations that do not correspond to any knot projection (as it take place, as an example, in the generator based on DT-code).

Obviously, using cascade representation we can construct a generator that satisfies the properties mentioned above. For a given projection with $2k$ legs the set of child projections is obtaining by adding one of the symbols Ψ , Σ , Σ in all possible $2k$ ways to a cascade diagram of the projection ($6k$ descendants total). The effectiveness of the algorithm can be enhanced appreciably using canonical cascade code.

Suppose we have the full set \mathcal{P}_n (with no repetitions) of projections with n crossings; where each projection is represented by its canonical cascade code. We can add the $(n+1)$ -th vertex (the $(n+1)$ -th layer) to a given cascade diagram from \mathcal{P}_n in $6k$ ways (where $2k$ is number of legs); so the diagram generates $6k$ descendants with $n+1$ crossings. Then the following simple rule can be used to decide if we should reject a given child projection or tabulate it.

If the $(n+1)$ -th vertex is not canonical root-vertex \longrightarrow reject the projection. (**)

For that first of all it should be checked if the new $(n+1)$ -th vertex belongs to the R -subset (1) or not. If “not”, we reject the projection without calculating the invariant root-code; if “yes”, and the R -subset contains more than one element, the invariant root-code should be found to identify the canonical root-vertex.

Thus, from the new generation of cascade diagrams only diagrams in canonical form will survive.

The deciding rule (**) ensures the following essential property. Because the canonical cascade code of a projection determines uniquely its genealogy (2), projections descend from different parents are necessarily different. So, in the sifting stage there is no need to compare projections obtained from different parents.

Moreover, we can get the set \mathcal{P}_{n+1} (projections with $n+1$ crossings) with no duplicates at all if we take into account information about symmetries of projections from the \mathcal{P}_n set.

First let us show that no asymmetrical projection can generate two equivalent canonical cascade diagrams.

Theorem 2. *Let P_0 be a projection with trivial symmetry group; C_0 is the canonical cascade diagram of P_0 and cascade diagrams C_1 and C_2 are descendants of C_0 (Figure 10).*

Let the projections P_1 and P_2 , which correspond to the cascade diagrams C_1 and C_2 , are equivalent. Then cascade diagrams C_1 and C_2 cannot be simultaneously canonical.

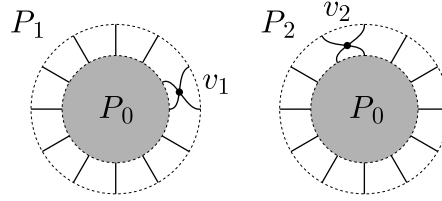


Figure 10 On the proof

Proof. Suppose, to the contrary, that cascade diagrams C_1 and C_2 are both canonical. Then, by the definition of canonical cascade diagram, both the vertices v_1 and v_2 (Figure 10) are canonical root-vertices. Now, let us note an obvious corollary of the canonical root definition (Def. 10): any transformation from the dihedral group D_{2k} that transfers a canonical root into a canonical root, transforms the projection into itself. Therefore, the (non-trivial) transformation σ that transfers the root v_1 into the root v_2 is in the symmetry group of the projection $P_1 \cong P_2$. But then, obviously, the symmetry group of the projection P_0 also contain σ . This contradiction completes the proof. \square

Thus, we see that only symmetrical projections can produce duplicates; however it is easy to avoid these duplications. Obviously, two child projections P_1 and P_2 of a given symmetrical projection P_0 are equivalent if P_1 can be transformed into P_2 (up to isotopy in the outside annular layer (see Figure 10)) by a symmetry of P_0 . Among all $6k$ child projections we should consider only projections that cannot be transformed one into another by a transformation mentioned above; let us denote this subset S . For example only two of 12 Ψ -child projections of projection P_0 (Figure 11) are in the S subset, namely the projections P_1 and P_2 in the Figure.

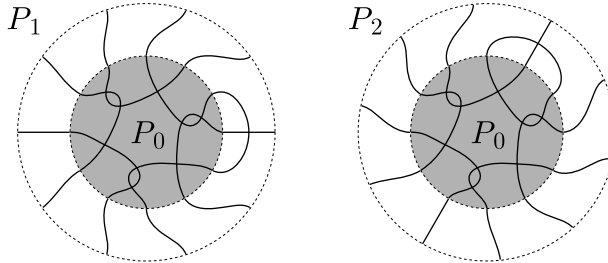


Figure 11 Child projections of a symmetrical projection

Similar to the asymmetrical case, it is possible to prove that if there are equivalent projections in the subset S , then only one of them can be in canonical form.

So, if the rule $(**)$ is used (taking symmetries into account) then the set of descendants of the set \mathcal{P}_n is exactly the set \mathcal{P}_{n+1} .

As evidenced from the above, any projection has a unique parent projection. Therefore, we can represent the set \mathcal{P} of all tangle projections as a genealogical tree starting from the one-crossing ancestor. In Figure 12 the first three stages of the tree are depicted. It is easy to demonstrate that the genealogical tree has no dead-end branches: any projection has at least one descendant.

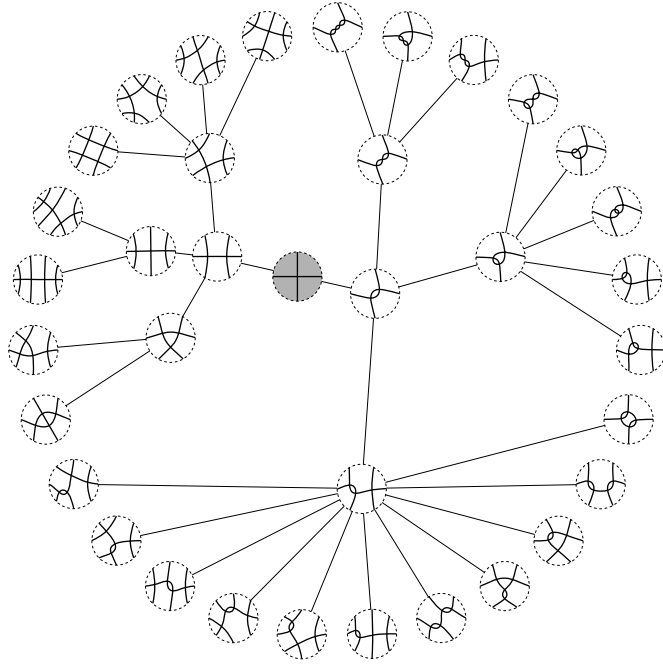


Figure 12 Genealogical tree of tangle projections

In conclusion let us describe the algorithm entirely.

Let \mathcal{P}_n be the full set of projections with n crossings. We need to do the following steps for each projection in \mathcal{P}_n to generate the set \mathcal{P}_{n+1} :

1. find allowable positions for symbols Ψ , X , Δ taking into account symmetry of the projection;
2. eliminate the cases when the new $(n+1)$ -th vertex is not in the R -set (1) of the child projection;
3. eliminate the cases when the child projection is composite;
4. eliminate the cases when the new $(n+1)$ -th vertex is not canonical root-vertex of the child projection;
5. add the remaining child projections to the catalogue;
- 6*. find flipping-equivalent projections, eliminate the duplicates.

We will not dwell here on the algorithm of flipping-equivalence checking. The algorithm we have used differs somewhat from the algorithm [13, 14, 15], we will describe it in a separate paper.

7 Tables of tangle projections

In this section we provide some results of enumeration of tangle projections and some important sub-classes of the set \mathcal{P} . We also give tables of alternating k -tangles with five crossings or less.

We use the following notation. As above, \mathcal{P} denotes the set of connected prime k -tangle projections. If we assume that projections related by a series of flipes (Figure 2) are equivalent, we obtain the set of alternating k -tangles denoted by \mathcal{A} . If we consider k -tangles up to weak equivalence (Def. 3), many projections became equivalent. For example, any projection that contains a boundary crossing with more than one legs are equivalent to a projection with smaller crossing number. We call such projections *weakly equivalent* and denote the set of weak equivalence classes by \mathcal{W} . Finally, we are interested in classification of k -tangle projections that have no more than one edge between any two vertices (do not contain 2-faces). We call projections of this type *reduced* and denote the corresponding set by \mathcal{R} . Reduced projections play a role analogous to the role of Conway polyhedra [1] in classification of knots and links.

Subsets of projections with fixed crossing number n and number of legs $2k$ we label by indexes. For example, $\mathcal{A}_{5,4}$ is the set of alternating tangles with 5 crossings and 8 legs; \mathcal{R}_7 is the set of all reduced projections with 7 crossings.

Using cascade representation and the algorithm described above we tabulate the sets of projections \mathcal{P} , \mathcal{A} , \mathcal{W} , and \mathcal{R} up to 12 crossings.

Table 1 contains numbers of projections in subsets $\mathcal{P}_{n,k}$ for $n \leq 12$.

Table 1 Number of tangle projections with n crossings and $2k$ legs

$k \setminus n$	1	2	3	4	5	6	7	8	9	10	11	12
2	1	1	2	6	19	71	293	1 348	6 568	33 701	178 706	973 085
3	.	1	2	8	29	138	638	3 237	16 805	90 239	494 151	2 756 453
4	.	.	2	8	41	210	1 125	6 138	34 112	192 278	1 096 560	6 317 363
5	.	.	.	5	31	231	1 458	9 183	56 084	340 885	2 060 224	12 446 400
6	16	161	1 406	10 572	74 331	499 902	3 276 104	21 112 641
7	60	840	8 818	75 747	591 091	4 327 816	30 451 898
8	261	4 702	56 199	541 570	4 628 641	36 633 417
9	1 243	26 753	361 106	3 846 580	35 758 786
10	6 257	155 593	2 332 512	27 199 662
11	32 721	916 595	15 123 600
12	175 760	5 464 661
13	963 900
all	1	2	6	27	136	871	6 021	45 241	352 856	2 839 086	23 333 649	195 201 866

For the set of alternating k -tangles along with the enumeration results (Table 2) we provide also tables of tangle pictures Figure 13 and Figure 14.

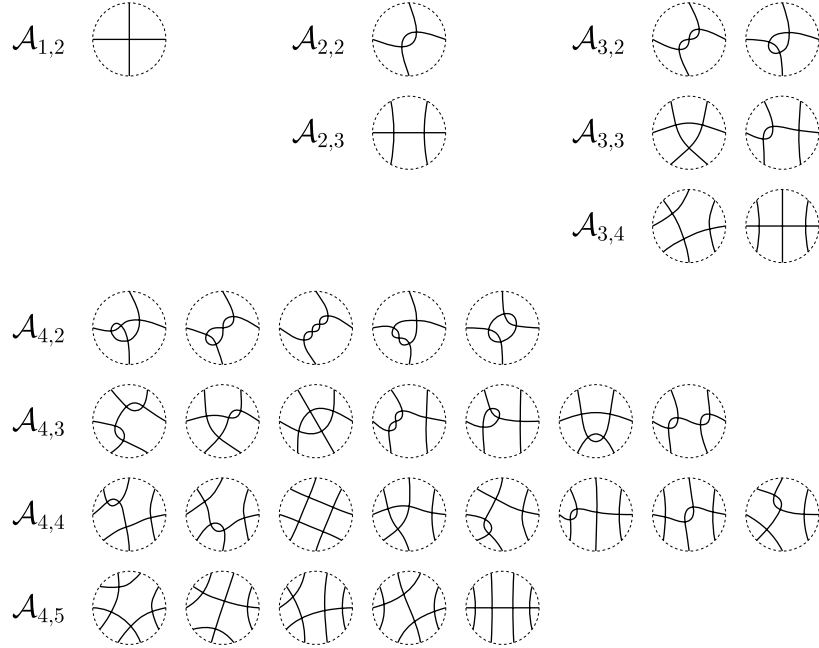


Figure 13 Alternating tangles with up to 4 crossings

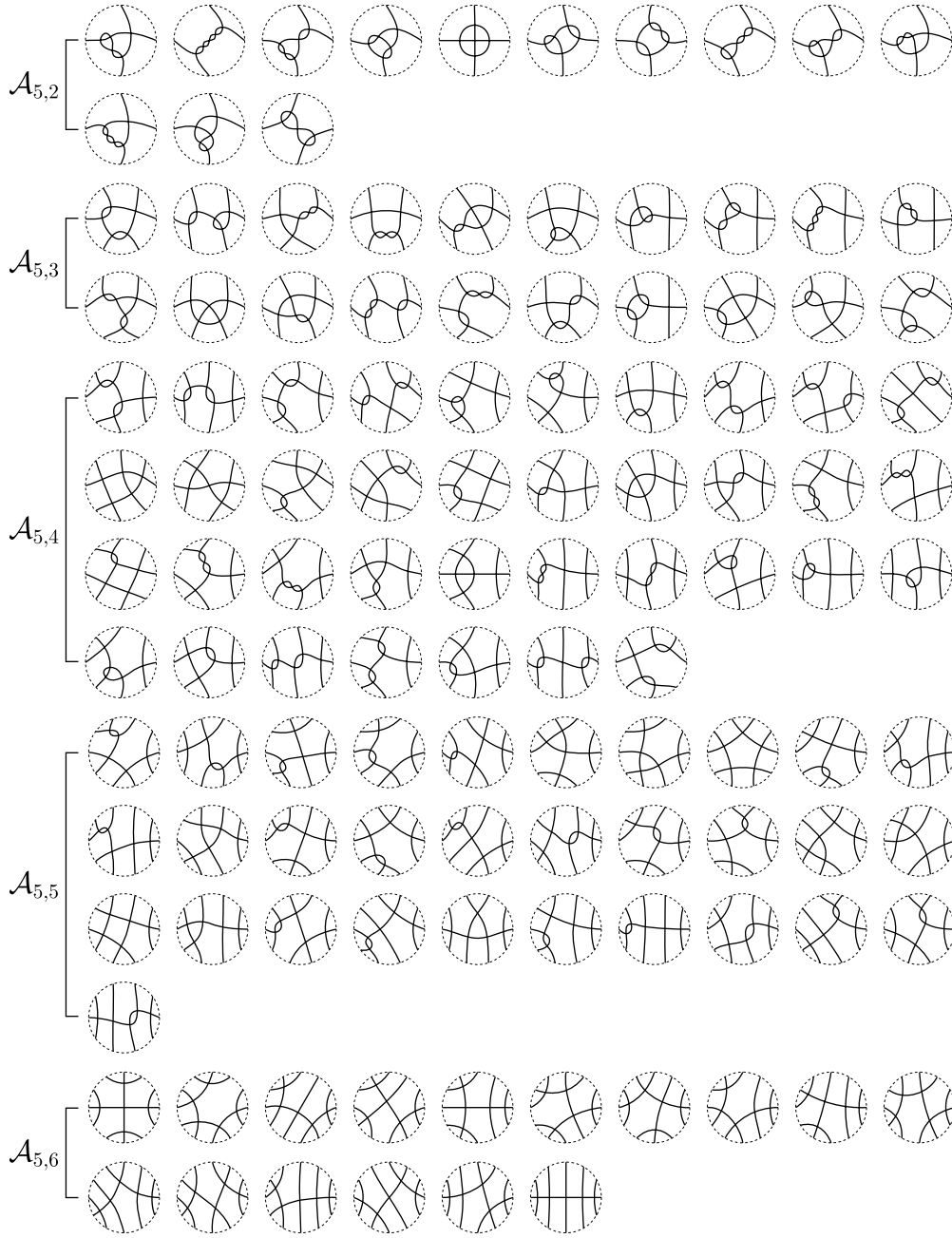


Figure 14 Alternating tangles with 5 crossings

Table 2 Number of alternating tangles with n crossings and $2k$ legs

$k \setminus n$	1	2	3	4	5	6	7	8	9	10	11	12
2	1	1	2	5	13	36	111	373	1 362	5 378	22 807	102 617
3	.	1	2	7	20	77	276	1 135	4 823	21 734	101 307	488 093
4	.	.	2	8	37	157	687	3 052	13 981	65 797	317 506	1 565 163
5	.	.	.	5	31	209	1 128	5 986	30 556	155 964	795 918	4 092 027
6	16	161	1 294	8 528	51 475	294 366	1 637 855	8 979 493
7	60	840	8 206	62 895	428 254	2 702 902	16 313 106
8	261	4 702	52 815	460 189	3 475 551	23 979 733
9	1 243	26 753	341 878	3 327 424	27 625 056
10	6 257	155 593	2 221 544	23 869 621
11	32 721	916 595	14 473 275
12	175 760	5 464 661
13	963 900
all	1	2	6	25	117	700	4 597	33 225	250 917	1 961 874	15 695 169	127 916 745

For the subset \mathcal{W} of weak equivalence classes of projections we obtain the following results.

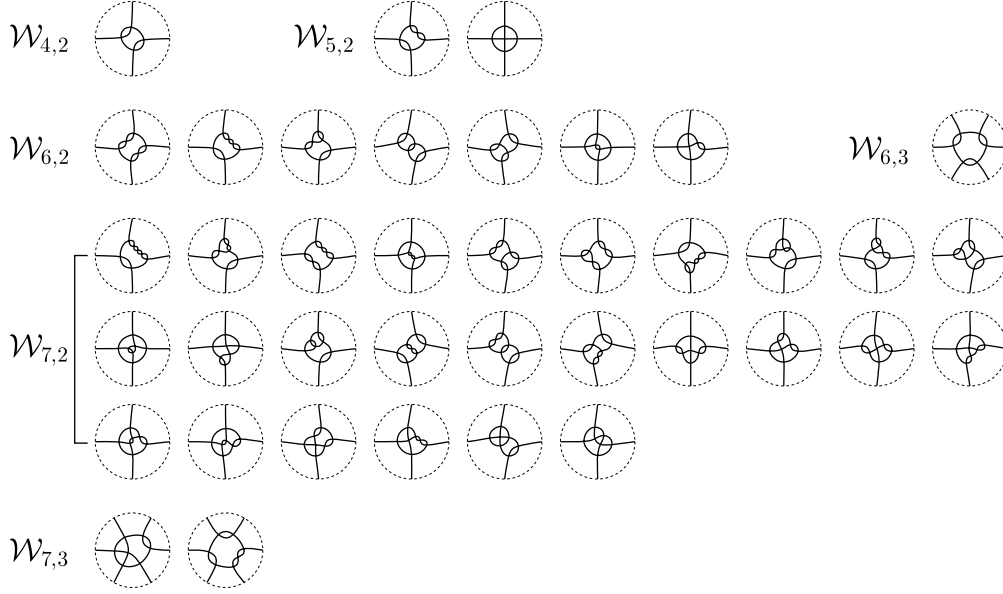


Figure 15 Alternating tangles with 7 crossings or less up to weak equivalence

Note, that this catalog agrees with the table presented in [3].

Finally, let us present the results for the subset \mathcal{R} of reducing projections.

Table 4 Number of reduced projections with n crossings and $2k$ legs

$k \setminus n$	1	2	3	4	5	6	7	8	9	10	11	12
2	1	.	.	.	1	1	3	9	26	74	238	770
3	.	1	1	1	1	4	7	24	69	226	719	2 423
4	.	.	2	2	4	7	21	58	185	596	1 998	6 753
5	.	.	.	5	9	22	49	152	458	1 545	5 188	17 990
6	16	42	126	355	1 144	3 769	13 012	45 515
7	60	228	799	2 586	8 850	30 754	109 843
8	261	1 288	5 164	18 745	68 142	248 891
9	1 243	7 525	33 856	134 834	520 884
10	6 257	44 482	222 482	962 620
11	32 721	266 270	1 464 500
12	175 760	1 607 405
13	963 900
all	1	1	3	8	31	136	695	3 928	23 414	144 864	919 397	5 951 494

References

- [1] Conway J. An enumeration of knots and links, and some of their algebraic properties. Computational Problems in Abstract Algebra (John Leech, ed.), Pergamon Press, Oxford and New York, 1969, 329-358.
- [2] Cromwell P. Knots and links. Cambridge: Cambridge university press. 2004. 350 p.
- [3] Kanenobu T., Saito H., and Satoh S. Tangles with up to seven crossings. Interdisciplinary Information Sciences, Vol. 9, No. 1, pp. 127-140 (2003).
- [4] Kauffman L., Lambropoulou S. On the classification of rational tangles. Advances in Applied Mathematics Volume 33, Issue 2, August 2004, Pages 199-237. e-print: [arXiv:math/0311499v2](#)
- [5] Sundberg C., Thistlethwaite M. The rate of growth of the number of prime alternating links and tangles. Pac. J. Math. **182** (1998) No.2 329–358.
- [6] Jacobsen J.L., Zinn-Justin P. The combinatorics of alternating tangles: from theory to computerized enumeration. e-print: [arXiv:math-ph/0111011v1](#)
- [7] Zinn-Justin P., Zuber J. Matrix integrals and the generation and counting of virtual tangles and links. J.Knot Theor.Ramifications 13 (2004) 325-356. e-print: [arXiv:math-ph/0303049](#)
- [8] Kauffman L. Virtual knot theory. European Journal of Combinatorics. 1999. 20. P.663–690. e-print: [arXiv:math/9811028v3](#)
- [9] Kuperberg G. What is a virtual link? Algebr. Geom. Topol. 3 (2003) 587–591. e-print: [arXiv:math/0208039v2](#)
- [10] Grishanov S., Meshkov V., Omelchenko A. Kauffman-type polynomial invariants for doubly periodic structures. Journal of knot theory and its ramifications. Vol.16. No.6 (2007) p.1–10.
- [11] Hoste J. The enumeration and classification of knots and links. Handbook of Knot Theory, W. Menasco and M. Thistlethwaite, eds., Elsevier (2005) 209–232.
- [12] Dowker C.H., Thistlethwaite M. Classification of knot projections. Topology Appl. 16 (1983), no. 1, 19–31
- [13] Rankin S., Schermann J., Smith O. Enumerating the prime alternating knots, Part I, Journal of Knot Theory and Its Ramifications, Vol. 13, No. 1 (2004) 57-100 e-print: [arXiv:math/0211346](#)
- [14] Rankin S., Schermann J., Smith O. Enumerating the prime alternating knots, Part II, Journal of Knot Theory and Its Ramifications, Vol. 13, No. 1 (2004) 101-149 e-print: [arXiv:math/0211348](#)
- [15] Rankin S., Smith O. Enumerating the Prime Alternating Links Journal of Knot Theory and Its Ramifications, Vol. 13, No. 1 (2004) 151-173 e-print: [arXiv:math/0211451](#)

